

第 1 章 概述

在软件行业里,架构师们的头上仿佛都带有光环。他们往往对复杂的问题举重若轻。几乎每一个年轻的程序员都希望有朝一日自己也能成为一名经验丰富的架构师,领导着一个开发团队、解决着世界上最复杂的软件架构设计和实施的问题。

然而,一名成功的架构师到底学习了哪些东西、又经历了怎样的历练,似乎没有人讲解过;大学里从来不曾开设过相应的课程,更没有人能够提供一张“课程表”;市面上的关于架构的图书大多或偏重于讲授抽象的设计原则,或偏重于对设计思想的感悟。读者如果没有亲身经历过具体的项目案例,抽象的设计原则缺乏系统的应用指导和可执行性,而感悟只有在读者亲自做过之后才有可能产生共鸣。那些缺乏经验的新人该怎么办呢?他们是多么希望有一张通向架构师的路线图啊!

1.1 什么是架构和架构师

万事开头难,文章开篇难!为了建立一个大家理解相同、不产生歧义的沟通基础,我们必须从两个最基本的概念入手。

首先,最重要的概念就是架构。按照维基百科的说法^①:

软件架构是指软件系统在高层次上的结构、创建此类结构的指导原则,以及这些结构的相关文档。这些结构可以用来推断和评价待建的软件系统。每一个结构包含软件的组成部分及其相互之间的关系,以及组成部分和相互关系的属性。一个软件系统的架构类似于建筑的架构。

① https://en.wikipedia.org/wiki/Software_architecture

其次,就是架构师的概念及分类。还是按照维基百科的说法^①:

软件架构师的工作就是进行高层次上的设计方案选择、制定相关的技术标准,包括软件编码标准,并确定所使用的软件工具和平台。

尽管有人将架构师的种类分得很细^②,实际最常见的架构师分为两种。

- 企业架构师(Enterprise Architect): 研究的对象是解决方案架构师在实施工作过程中所使用的方法,为后者解决具体的业务问题提供架构设计以及实施的具体步骤和方法指导。
- 解决方案架构师(Solution Architect): 实际承担解决企业业务问题的任务。有可能需要使用企业架构师所提供的架构设计以及实施的具体步骤和方法指导。

换句话说,企业架构师解决的是 IT 问题,而解决方案架构师解决的是业务问题。贯穿本书所指的架构师是后一种,即解决方案架构师。不仅如此,本书面向的是那些解决方案涉及多个功能系统的使用、架构原则和思想具有横跨企业的指导意义的架构师。

1.2 这本书是为谁写的

本书针对的读者群包括希望成为解决方案架构师的程序员、IT 咨询师,希望通过与同行进行交流而得到提高的架构师,还有希望了解如何能够让自的部门有效地应对不断变化的企业业务要求的各级 IT 领导。

一名 IT 从业人员可能正处在下面列出的一种情形之中:

1) 刚刚走出大学的校门、参加工作。在计算机系里已经学会了一门或几门编程语言(如 Java、C#、Python,等等),以及数据结构和算法,对后台数据库、网站架构甚至 SOAP Webservices 都有初步的了解,并且可以很熟练地进行编程来解决别人交给的非常具体的问题。但是如果面对类似本章 1.3 节中所描述的那几个实战例子就不知从何下手了。

2) 从事软件开发工作 3~5 年,十分胜任小型或局部问题的分析、方案设计和具体实施。然而面对规模稍大、更加复杂并涉及多个系统的业务问题的设计

① https://en.wikipedia.org/wiki/Software_architect

② <https://blog.prabasiva.com/2008/08/21/different-types-of-architects/>

任务时会感到力不从心,不知道从何下手,不知道应该采用什么样的原则以及设计和实施步骤,也不知道应该使用何种工具。

3) 作为一名具有 2~3 年实际经验的架构师,已经参与和主持了几个系统集成项目的设计和实施工作,但对为什么采用某个设计方案、其优点和缺点的评估却说不出个所以然来,因此无法在下一个项目的工作中信心十足地再次采用类似的方案。

4) 从事架构师的工作已有 5~10 年,能够深入了解具体设计方案背后的来龙去脉,以及设计方案的优点、缺点甚至相应的补救措施。然而,面对一个复杂项目各方面的利益相关人(如项目出资方、业务分析人员、其他设计人员、开发团队、项目经理、合作伙伴,等等),深深地感到将项目设计的思想和方案优缺点论述清楚并得到方方面面的支持是一件十分困难的事情。即便是开发团队内部的技术细节的沟通和统一也不是那么容易。

5) 作为一名具备多年实践经验的企业 CIO,面对行业内竞争、行业外颠覆的压力,以及企业业务对 IT 能力的要求与 IT 部门实际交付能力之间日益增大的差距(如图 1-1 所示)深感忧虑(其背后的直接原因包括移动设备、云计算、社交网络、大数据、物联网等的广泛和深入的使用),并苦苦探索可从 IT 技术和企业组织结构的不同角度对这个日益严重的问题做出有效的反应。

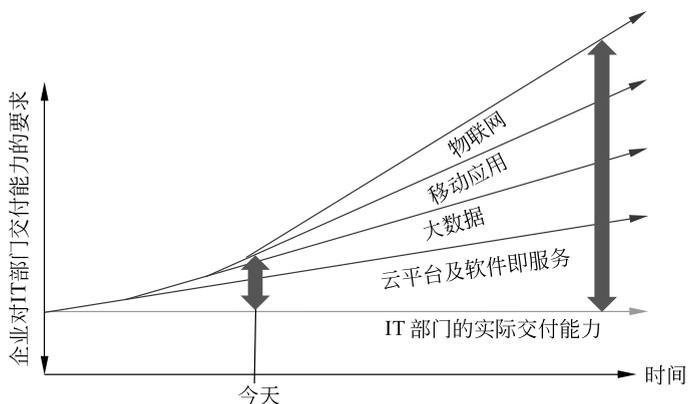


图 1-1 数字时代的压力造成了 IT 部门“欠债”越来越多

1.3 为什么写作此书

在从事软件服务工作的近 20 年中,作者亲身经历了北美 18 个行业、50 多个客户的大型项目,其中还包括两三个完全失败的项目。客户中的绝大多数是

财富 500 强的公司,有些甚至是 50 强。

我们先来粗略地感受一下这些项目。

- 某跨国石油公司的能源交易部门于 2002 年希望建立一个全球范围的能源产品(包括石油、天然气、电力、污染排放指标等)交易结算和风险评估平台,能源产品可在全世界各地进行交易,而新的交易实时地与交易伙伴进行结算,本部门所持有的能源产品组合的细节及其系统风险被实时地更新。如果交易的一方或双方隶属于某个母公司,母公司本身所持有的能源产品组合的细节及其系统风险也被实时地更新,任何过度风险的情形都会被及时预警。
- 美国某著名的航空快递公司在 2005 年寻找一个高效、耐用的消息(message)交换平台,能够每天交换 86000 万条在邮件递送过程中由每一个处理中心的进出扫描而产生的消息。而这些消息在全天的任何时间、在世界上的任何地方不断地产生着。在此平台上需要进一步构建供广大消费者使用的邮件跟踪服务。到 2010 年,这个平台的消息总处理能力达到了每天 50 亿条。而且以不同名称独立运营的同一个母公司下的所有分公司使用快递服务的总量,以及使用的模式可以被累加出来,为下一年针对不同母公司的大客户进行更好的服务定价提供可靠的基础。
- 美国某知名零售连锁店有 800 多家店面,销售包括衣服、家电、日用品等在内的上万种商品。每个星期,总店都会在不同的商品上推出各种不同的促销折扣以及减价券,而有些时候这些促销折扣和减价券的有效期只有某一天中的几个小时。促销折扣及减价券的推出上线必须经过特定权限的批准,在此过程中完全不允许出现停机现象。
- 美国某玩具公司拥有多个世界上最著名的玩具品牌。在 2012 年引入了产品生命周期管理 PLM(Product Lifecycle Management)的软件系统后,PLM 成为玩具产品设计、制造、市场开发、销售、服务等几乎所有的企业职能部门共同需要的平台。然而,由于历史的原因,这些职能部门目前使用的外购和自己开发的应用系统超过了 50 个,加上这些职能部门分布在北美、欧洲、中国、东南亚等多个地区或国家,实际上根本不可能让这些职能部门全面放弃已经使用了十几年甚至几十年的、他们所熟知的系统。然而,由于所有围绕玩具产品的各类职能的数据已经全部转移到 PLM 平台上,如何在保证这些职能部门继续使用现有系统的同时,能够从新的 PLM 平台上及时、有效地获得不断更新的相关数据?

- 美国某著名的二手车销售连锁店在美国和加拿大拥有近千家店面,销售经过重新认证的二手车。总店拥有自己的 IT 系统,而每一个分店除了可以调用总店的 IT 系统外,还有一套本地备份,以便本地系统在与总店 IT 系统失联的情况下仍能应付日常工作。连锁店在 2010 年面临的挑战之一就是,每当总店推出一个新的系统部署时,比如升级和更新,甚至仅仅只是网页上的横幅图标按某个节日进行临时的更换,如何能够最有效地部署到所有近千个店面的本地 IT 系统中,同时了解哪些店面的部署出现了错误并进行妥善的处理。这是一个相当具有挑战性的问题^①。

仔细研究一下上面这几个实战案例,我们也许会在某个局部发现书本上学过的 N 层网站架构或者某个设计模式(Design Pattern),但组合起来造成的问题之复杂,是我们从前不可想象的。

环顾软件以外的其他行业,对过去的相关案例进行深入分析和总结,无论对个人还是团队来说,都是非常有效的学习手段。在商学院、医学院、法学院、军事学院和体育学院的课程安排里,案例的分析是学习的内容和过程中最重要的、也是最引人入胜的部分。通过例子来学习、先模仿再深入理解是最有效的学习方法之一。在作者的软件咨询服务工作过程中,客户最享受的部分就是“听故事”,然而作者却没有在美国和中国任何一所高校的计算机系网站上公布的授课内容中找到软件项目设计和实施的案例。

除技术因素外,作者在主持项目设计和实施的过程中,对如何与客户、合作伙伴,以及自身领导的开发团队成员进行架构设计指导思想的沟通与说服工作,也积累了相当的感悟。

以上两点促使作者下定决心,抛砖引玉,力图提供一条由程序员到架构师的路线图,并结合实战的架构设计案例来对抽象的设计原则进行展开说明,为希望成为架构师的程序员们的学习和实践过程进行具体的指导。本书将采取理论阐述和动手开发相结合的方式,以保证学习和能力提高的质量。

1.4 通往架构师之路的路线图

针对上面列举的通往架构师的道路上的不同阶段,本书拟引入如下的路线图。

^① 我们最终的多店面部署解决方案被称为 Store-in-a-Box。

1) 起始于相互独立的系统,首先讨论如何进行系统集成(第2章~第4章)。具体内容包括系统之间进行集成时常见的相互作用模式、被集成系统的功能分类,等等。这一部分的内容已有大量的书籍进行了各种深入的阐述。然而在实际项目中遇到的许多关于设计方案选择的把握并未见有论述;对系统集成设计的体会和设计思想上的领悟也未见有太多公开的发表。因此,希望对系统集成和服务架构有一定经验和体会的读者也不要轻易跳过这几章。作者真心地期待你使用前言末尾列出的电邮地址分享案例、体会和领悟,抒发情怀。

2) 在通过集成、系统与系统之间的连接初见成效之后,引入服务的概念(第5章),并对围绕服务的项目实施具体工作内容进行了解(第6章、第7章)。服务概念的出现虽然已有至少15年的历史,但在绝大多数实际的项目实施过程中服务常常沦为“点对点”实施的一种新的连接机制,而整个架构思想换汤不换药。其结果是,根本性的技术问题依然存在,一个也没有彻底解决。

3) 在积累了一定的系统集成和服务项目的经验后,第8章承上启下,以作者在近20年的实践中对系统集成和服务的方法以及在技术层面和IT组织结构层面上的局限性的总结和思考,引入现代API的概念,并与围绕服务和系统集成项目的实施进行对比,从而对最新的、围绕API的架构理念有一个初步的认识。

4) 了解围绕API进行解决方案开发工作的具体内容,并对具体实施方法背后的深层思想进行梳理(第9章、第10章)。不论采用什么样的API的设计和开发工具,这个过程大致相同:从API的提供方看,涉及API开发生命周期中的各种活动;而从API的使用方看,则涉及如何发现并正确使用API。

5) 深入了解API与微服务(以及服务)之间的关系(第11章)。这个问题常常被客户问到,并且具有理论和实践上的重大意义。

6) 深入了解API的部署方式,特别是正在兴起的云端部署方式(第12章)。云端部署为API以及集成应用的目标环境提供了一种新的选择。API架构师的目标是采用完全不依赖于目标环境(比如本地/数据中心、云端的虚拟机、云端带有负载平衡器、集群等相关设施的部署环境,以及以上各种类型的混合体,等等)的基础代码,而是通过因目标环境而不同的配置上的变化来实现不同环境下的部署。

7) 在上述学习和提高的过程中,不断积累最佳实践的经验和教训(第13章),了解新的架构思想对企业业务发展的影响(第14章),加深在这方面的认识。

如果要打个比方,上面提议的路线图有点儿类似于一个从士兵到将军的计划。士兵的责任在于提高体能,掌握军事技能和武器装备;中层军官的责任在于

熟悉自己权限以内的兵力调动,指挥战斗和局部的战役;而高级将领的责任则在于熟知军事服务于政治,时刻牢记进行战争的最终目的,有能力指挥各兵种及各级军官并协调友军进行大规模的立体作战。

1.5 架构师应该具备的素质

要想成为一名优秀的架构师,除了高超的计算机软件专业方面的知识以外,还必须具备一定的“软实力”。有些软实力看似十分简单和基本,但在具体的执行过程中常常被遗忘和忽略。而错误的发生往往就是因为人们忽略了最简单明了的一些基本原则。

- 永远把解决客户的业务需求放在第一位。IT 技术是手段,不是目的。无论你掌握的 IT 技术有多先进、多“酷”,如果不能解决客户具体的业务问题,你掌握的技术就会被客户看成是一无是处。所以作者经常讲的一句话就是,“架构师别太把自己当回事儿”。
- 超强的逻辑性。这其中既包括分析问题的数理逻辑能力,也包括在阐述论点和设计思路过程中的一致性、连贯性和洞察力。
- 永远开放的头脑。倾听和认真分析各种意见,始终抱着一种将事情做到极致的决心。
- 广泛的知识面,对 IT 技术和业务知识有着同样浓厚的兴趣。如果对所要解决的业务问题漠不关心,是不可能完善地使问题得到解决的。
- 超强的学习能力,并学以致用。学习的内容包括 IT 技术、业务知识、管理知识、认知科学甚至心理学等人文方面的知识。
- 注重结果。无论开始和过程有多么华丽,只有结果的辉煌才是真正的成功。架构师工作的成功来自于项目的圆满完成、用户预期从项目成功中获得的价值得到实现。

1.6 对架构师的学习和培养过程的几点建议

除了以上阐述的成为架构师的路线图,以及作为一名合格的架构师所应有的基本素质之外,作者对有志成为架构师的 IT 人士还有以下几点建议。

- 在学习过程的最开始,明确说出作为程序员或者初级架构师在工作中所面临的困惑,并记录下来。在今后学习的过程中,不时拿出这些困惑来再读一读,看看是不是有的困惑已经得到了解决;同时记录下新的困惑和问题。
- 在学习过程的最开始以及阶段性的开始,明确说出学习和实践试图达到的目标。这些目标必须是十分具体的,能在事后客观地进行衡量看是否达到了,并根据不断提高的现有认识水平提出更高层次的目标。
- 针对个人的具体情况,按照本书的建议列出为了达到目标所要学习的相关内容,对上面建议的路线图进行个性化的丰富和完善。
- 下棋要找高手。和其他的架构师进行交流,尤其是在特定的设计原则和方法及其实际应用上进行交流,对于一个架构师的成长和提高十分有必要。而这个过程会很有收获,也可以是很快乐的。
- 要使用合适的工具。如果你也认为仅仅带上装有榔头、钳子和改锥的工具包是不可能建成摩天大厦的,那么你就肯定会同意,必定需要合适的、贯穿软件生命周期各个阶段的、成熟的软件工具,才有可能完成大型复杂系统的架构设计和具体实施。

学习、实践、总结、提高,这才是成为一名合格的架构师的必经之路。

1.7 本书的主要内容

本书由 3 部分组成。

- 第 1 部分介绍系统集成架构的基础,并对系统集成与面向服务架构(SOA)实施细节的各个方面进行介绍,理论与实践并重。
 - 第 1 章 如何成为一名架构师:首先与读者一起建立我们讨论的共同起点,指出成为一名合格架构师的方向和路线草图,并初步勾勒出合格架构师必备的素质。同时,指导读者设立一种能够进行系统集成开发的技术环境。
 - 第 2 章 为什么要进行系统集成:对系统集成的历史、必要性、大原则及实施预后进行初步的讨论。
 - 第 3 章 系统之间相互作用的模式:主要探讨参与集成的各个系统之间相互作用的方式、适用范围及其优缺点。

- 第4章 常见的参与集成的功能系统：列举常见的参与集成的系统本身的功能，比如数据库、客户管理系统(CRM)、企业资源计划(ERP)系统、BPM、复杂事件处理(CEP)等，以便读者对经常碰到的、需要进行集成的系统有一个大致的了解。
- 第5章 究竟何为服务：这是一个老话题，但常常没有说清楚。
- 第6章 系统集成项目的实施步骤：介绍典型的系统集成项目的具体实施细节，包括整个生命周期的各个环节。
- 第7章 具体项目与公共服务：介绍如何将每个具体项目都需要使用的普遍性的服务模块单独进行开发，并利用标准化的项目模板来对每个具体项目进行实施，从而避免公共服务功能部分的重复实施，让每个项目专注于解决各自具体的业务问题。公共服务除了与业务有关的部分外，还包括安全、监视和管理以及运行维护方面的内容，但这部分内容不是本书的重点，仅仅是围绕重点涉及到的话题，所以点到为止。
- 第8章 SOA 的实施、局限性及解决方法：回顾 SOA 应用十几年来所产生的效果，分析其局限性，以及相应解决方法的展望。
- 第2部分在第1部分的基础上引入现代 API 的概念，并就 API 对于企业业务发展的意义、围绕 API 开发工作的具体细节、API 与其他相关技术的关系等，结合实践进行详细的阐述。
 - 第9章 现代 API 的引入及应用互联网的概念：介绍 API 的概念、使用 API 后企业对业务和 IT 可期待的愿景，以及 API 与系统集成的关系。
 - 第10章 围绕 API 的开发工作的内容：详细介绍 API 开发和应用中的技术细节、以 API 为主导的架构设计，以及对企业 IT 部门与业务部门之间的互动带来的影响。
 - 第11章 API 与微服务：从理论和实践的角度论述 API 与十分流行的微服务之间的联系与区别。
 - 第12章 API 与云计算：对 API 的部署环境，特别是目前迅速兴起的云计算环境，以及 API 的部署模式进行介绍。
 - 第13章 最佳实践的经验：对开发和应用 API 的过程中积累下来的经验及教训进行总结和概括。
 - 第14章 API 经济：当每一个企业以及企业里的每一个项目都按照

API 的架构思想进行实施,业务资源以 API 的形式系统地呈现时,就会形成一个应用网络(Application Network)。而这个网络也是企业价值链的网络,即以 API 支撑的经济体。这一章将对 API 经济进行初步的介绍,并引入企业数字化转型的话题。

- 第 3 部分是技术以外的随感。
 - 第 15 章 架构师的人文情怀: 这部分内容天马行空,对技术方面的感悟、与人沟通的软实力、架构师的教育和职业规划,甚至学习过程的分析等都有涉及,十分随意。

1.8 总结

本章首先澄清了什么是架构,什么是架构师。然后对不同阶段和不同角色的相关人员目前就大型复杂系统架构的理解所处的状态进行了分类,作为学习过程中不同阶段的代表。

随后,本章列出了一份通往架构师之路的路线图简介,同时还抛出了作者眼里优秀架构师必须具备的素质,以及针对成为架构师的学习过程的几点建议。本书各章的内容安排也是按照这个路线图展开的。

和其他很多技能一样,成功地成为一名优秀的架构师必须通过实践,没有“捷径”可走。本章最后选用(而不是推荐)了可以免费获得的一个系统集成及 API 的开发和部署平台,供读者练手,并得以对抽象的设计原则利用实例进行具体的说明。

在这个历程的终点回报丰厚,而过程本身也可以是充满乐趣的。你准备好了吗?